An Implementation of Markov Regime Switching GARCH Models in Matlab

Thomas Chuffart

Aix-Marseille University (Aix-Marseille School of Economics), CNRS & EHESS

Abstract

MSGtool is a MATLAB toolbox which provides a collection of functions for the simulation and estimation of a large variety of Markov Switching GARCH (MSG) models. Currently, the software integrates a method to select the best starting values for the estimation and a post-estimation analysis to ensure the convergence. The toolbox is very flexible a user-friendly with a large number possible options. In this paper, we give some illustrative examples.

JEL Classification Numbers:

Key words: Markov Switching GARCH models, Conditional volatility, Software tools, QMLE

1 Introduction

The purpose of this document is to introduce the user to the functionality of the Markov Switching Generalized Autoregressive Conditional Heteroskedasticity (MSGARCH) toolbox or **MSGtool**[1] package for MATLAB.

Regime Switching GARCH models belong to a class of models that yields the well known univariate GARCH models introduced by Bollerslev (1986) as a special case. The idea is that the volatility is characterized by regime switches driven by a Markov chain. Although attractive, there are copious empirical evidences in the econometric literature that argue against the suitability of the traditional GARCH model. For example, these models do not adequately fit the data over a long period of time. Lamoureux and Lastrapes (1990) show that if structural changes are not considered, it may bias upward GARCH estimates of persistence in variance. Thus, MSGARCH models can be useful. They have been introduced in time series analysis by Hamilton (1989) and are now very popular among econometricians. These processes give rise to a conditional mixture distribution, where each component is endowed with its own GARCH structure (see Haas and Paolella (2012)). Moreover, they allow a time-varying skewness as recommended by Rockinger and Jondeau (2002) contrary to traditional GARCH type models.

Economic intuitions can often be mapped to these MSGARCH models. It is a strong assumption to say that the volatility of an asset follows the same dynamics along the time. Structural changes introduce switches in the dynamic of these assets as shown by Lamoureux and Lastrapes (1990). Markov Switching GARCH models are rather flexible and have been found to fit asset returns well. This class of model is useful to model the time varying volatility where different states of the world affect the evolution of a time series. The dynamic depends on the present regime. This latter one is a realization of an hidden Markov chain with finite state space. A lot of empirical applications have used this kind of models, for example Hamilton and Susmel (1994), Brunetti et al. (2008) among others.

To our knowledge, even if these models are now very popular, there is no statistical software which is enough general, flexible and user-friendly. OxMetrics and PcGive can estimate MSGARCH processes. However, the choice of the form of GARCH components is very limited. Some functions are also available to estimate MS-Autoregressive models, for example Perlin (2015) and Ding (2012) propose a package to estimate MS-Vector Autoregressive models with Matlab. In this appendix, we present a very general toolbox and give some illustrative examples on financial returns. The user can simulate and estimate three MS-GARCH models with many options: the models of Gray (1996), Klaassen (2002) and Haas, Mittnik, et al. (2004) [2]. He has also the possibility to choose the distribution of the errors and the estimation method in a simple way. Our code is optimized to be used with the Matlab compiler and mex functions.

The paper is organized as follows. In section 2, we give a brief exposition on the topic of regime switching GARCH models. Section 3 describes the available functions and their particularities. Section 4 gives some illustrative examples.

[1]The package is still under development to accommodate new features. The up to date version can be downloaded from http://www.thomaschuffart.fr/codes-matlab/

[2]The MSGARCH model of Augustyniak (2014) will be implemented in the next release.

2 MS-GARCH models

(ε_t) is a MSGARCH process if, for $t = 1, \ldots, T$ with T the sample size, we have:

$$y_t = \varepsilon_t$$

with

$$\varepsilon_t = \eta_t \sqrt{h_t(\Delta_t)},\ \eta_t \sim IID(0,1)$$

and there exist $\alpha_0(\Delta_t)$, $\alpha_i(\Delta_t)$, $i = 1, \ldots, q$ and $\gamma_l(\Delta_t)$, $l = 1, \ldots, p$ such that

$$h_t(\Delta_t) = \alpha_0(\Delta_t) + \sum_{i=1}^{q} \alpha_i(\Delta_t)\varepsilon_{t-i}^2 + \sum_{l=1}^{p} \gamma_l(\Delta_t) h_{t-l}. \tag{1}$$

η_t is an identically and independently distributed random variable with zero mean and unit variance. Δ_t is a variable which indicates the state of the world at time t and follows a Markov chain with finite state space $S = 1, ..., k$, and a transition matrix P. Thus, the probability to switch from one regime to another depends on the transition matrix P, given by

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1k} \\ \vdots & \cdots & \vdots \\ p_{k1} & \cdots & p_{kk} \end{pmatrix}$$

with $p_{ij} = p(\Delta_t = j | \Delta_{t-1} = i)$ the probability to be in state j at time t given to be in the state i at time $t-1$. In this form, the model can not be estimated by QML since the calculation of the likelihood function for a sample of T observations is infeasible. It requires the integration of k^T possible regime paths where k is the number of regimes (Hamilton and Susmel (1994) and Cai (1994)). To circumvent the path dependence problem, Gray (1996) introduces a MS-GARCH model under the hypothesis that the conditional variance at any regime depends on the expectation of previous conditional variances. He proposes to replace h_{t-1} by the conditional variance of the error term ε_{t-1} given the information up to $t-2$:

$$h_t(\Delta_t) = \alpha_0(\Delta_t) + \alpha(\Delta_t)\varepsilon_{t-1}^2 + \gamma(\Delta_t) \sum_{i=1}^{k} p(\Delta_{t-1} = i | \Omega_{t-2}) h_{i,t-1}. \tag{2}$$

$h_{i,t}$ is the conditional variance in state i at time t, Ω_t is the information set of the process (i.e. the return history up to date $t-1$) and $p = q = 1$. Klaassen (2002) enlarges the information set up to $t-1$ by conditioning the expectation of previous conditional variances on all available observations and also on the current regime:

$$h_t(\Delta_t) = \alpha_0(\Delta_t) + \alpha(\Delta_t)\varepsilon_{t-1}^2 + \gamma(\Delta_t) \sum_{i=1}^{k} p(\Delta_{t-1} = i | \Omega_{t-1}, \Delta_t = j) h_{i,t-1}. \tag{3}$$

The model of Haas, Mittnik, et al. (2004) contrasts with this approach because each specific conditional variance depends only on its own lag,

$$h_t(\Delta_t) = \alpha_0(\Delta_t) + \alpha(\Delta_t)\varepsilon_{t-1}^2 + \gamma(\Delta_t) h_{t-1}(\Delta_t). \tag{4}$$

This model can be rewritten in matrix form:

$$\mathbf{h_t} = \boldsymbol{\alpha}_0 + \boldsymbol{\alpha}_1 \varepsilon_{t-1}^2 + \boldsymbol{\gamma h_{t-1}},$$

where $\boldsymbol{\alpha_0} = [\alpha_{01}, \alpha_{02}, ..., \alpha_{0k}]'$, $\boldsymbol{\alpha_1} = [\alpha_{11}, \alpha_{12}, ..., \alpha_{1k}]'$ and $\boldsymbol{\gamma} = diag(\gamma_1, \gamma_2, ..., \gamma_k)$. $\mathbf{h_t}$ is thereby a vector of $k \times 1$ components. These MS-GARCH models[3] can be easily estimated by Maximum Likelihood (ML) estimation following the work of Hamilton (1989). A ML estimation of θ_0, the parameter vector $\theta_0 = (\boldsymbol{\alpha_{00}}, \boldsymbol{\alpha_{01}}, \boldsymbol{\gamma_0})'$ to be estimated, is defined as

$$\hat{\theta} = arg\,max\,\mathcal{L} = \sum_{t=1}^{T} \log f(\varepsilon_t|\Omega_{t-1})$$

where $f(\varepsilon_t|\Omega_{t-1})$ is the conditional density of ε_t given the process up to time t. This density is the sum of conditional regime densities weighted by the conditional regime probabilities $Pr(\Delta_t = j|\Omega_{t-1}, \theta)$: The vector of parameters has to respect some usual constraints: the stationarity of the process and the positivity of the variance (see Haas and Paolella (2012) for details).

3 Overview of the toolbox

The MStool package is written for the simulation and estimation of a large variety of MS-GARCH models. The main functionality of the code is build around two functions: `swgarch()` and `swgarch_sim()` that we present in this section. Moreover, we list all the functions in Tables 1 and 2 with a brief description and their functionalities. The installation of the package is quite straightforward. In order to use the main functions, all you need to do is to tell MATLAB to place the files from the m Files.

3.1 Simulation

The `swgarch_sim` function returns the simulation of MSGARCH process. The user can specify both the type of process and the distribution error. This function simulates a Markov Chain which represents the true state of the nature. This latter one is not observed by the econometrician. We propose two different Markovian regime switching generation processes. In the first one, the conditional variance regimes are dependent. In the second one, the regimes are independent. In this last case, if a switch occurs, we move directly in an other regime. We consider that the past conditional variance was already in this regime. By construction, the estimation procedure proposed by Gray (1996) and Klaassen (2002) is expected to fit well the first type of switching whereas Haas, Mittnik, et al. (2004)'s model is expected to fit better the second one. The user calls the function with

- `datasim = swgarch_sim(dim, k, parameters, M, error_type, ms_type, param_dist, fig)`.

The function returns a structure `datasim` and a figure object `fig` if asked by the user. The structure encompasses five elements: ε_t (`datasim.vE`) the simulated residuals, h_t

[3]There are a number extensions of these two types of MS-GARCH processes. For example, Gallo and Otranto (2015) introduce asymmetric effects in each regime variances.

(`datasim.vH`) the simulated conditional variance, $\mathbf{h_t}$ (`datasim.mH`) the regime specific conditional variances and the true state of the nature (`datasim.mS`) simulated by the Markov chain. The figure `fig` represents both ε_t and h_t. `swgarch_sim.m` calls a routine `markovSim.m` which generates Markov chains. We present working example in Section 4.

3.2 Estimation

To estimate a MSGARCH model among the proposed ones, the user has to call the main function `swgarch.m`. This function calls many subroutines to speed up the estimation. Some of them can be translated in mex with the MATLAB coder toolbox. The function is coded to be very intuitive and flexible for the user. To estimate a MSGARCH model the user has to call:

- `[estimation, probabilities, diagnostics] = swgarch(data, k)`

where `data` is the residual vector and k the number of regimes. By default, this function estimates a MSGARCH model of Haas, Mittnik, et al. (2004) by ML assuming Normal distribution with an unconstrained optimization. A display message will ask the user how many starting values he wants to try to initialize the likelihood. Some options are also available:

- `[estimation, probabilities, diagnostics] = swgarch(data, k, error_type, ms_type, estim_cons, startvalopt, startvalG, startvalM, startvalDist).`

`error_type` is the error distribution, currently two distributions are available: `'NORMAL'` for the Gaussian and `'STUDENTST'` for the Student distribution. `ms_type` is the type of MSGARCH, it can be either `'HAAS'`, `'KLAASSEN'` or `'GRAY'`. `estim_cons` is to set up the kind of optimization, unconstrained or constrained. In the unconstrained case, we transform the parameters with the routines `swgarch_transform` and `swgarch_itransform` to respect the constraints on parameters. Then `swgarch` calls the `fminunc` optimization function. In the constrained case, `swgarch` calls the `fmincon` optimization function with `swgarch_constr` for the inequality constraints. `startvalopt` has to be a structure with two elements. The first element is `'YES'` or `'NO'` if the user wants to provide starting values or not. If he provides starting values, the second element can be empty ; in the other case, the user has to provide the number of sets to test before launching the optimization. `startvalG` is a matrix $k \times 3$ which contains GARCH parameters starting values and `startvalM` is a matrix $k \times k$ of the transition probabilities starting values. Finally, `startvalDist` is the starting value of the distribution parameter if `error_type` is not `'NORMAL'`.

The ouput is composed of three structures: `estimation`, `probabilities` and `diagnostics`. `estimation` is composed of the following elements:

- `estimation.garch`: the estimates of GARCH parameters.
- `estimation.M`: the estimates of the transition probabilities matrix.
- `estimation.VCV`: the variance covariance matrix.
- `estimation.VCVr`: the robust variance covariance matrix.

- `estimation.H`: the conditional predicted variance.

`probabilities` is composed of three elements: `probabilities.predict_proba`, `probabilities.smoothed_proba` and `probabilities.uncond_proba`. The first one is the predicted probabilities resulting from the likelihood estimation, the second one is the smoothed probabilities following Kim (1994) and the last one is the unconditional probabilities. Finally `diagnostics` contains usual diagnostic elements of the optimization like exit flag, numerical scores, hessian, the value of the likelihood etc.

4 Illustrations

4.1 Monte-Carlo experiments

In this section, we give some examples and show the performance of the toolbox using some Monte-Carlo experiments. We perform two experiments: in the first one we simulate data following a path-dependence MSGARCH and we estimate the MSGARCH of Klaassen (2002). In the second one, we simulate data following a non path-dependence MSGARCH and we estimate the MSGARCH of Haas, Mittnik, et al. (2004).

The set up of the experiments is as follow. We simulate two regimes. The parameters for the data generating processes (DGP) are $\boldsymbol{\alpha_0} = (0.1, 0.01)'$, $\boldsymbol{\alpha_1} = (0.1, 0.05)'$ and $\boldsymbol{\gamma} = (0.92, 0.7)'$ if we simulate a path-dependence MSGARCH or $\boldsymbol{\gamma} = \begin{pmatrix} 0.92 & 0 \\ 0 & 0.7 \end{pmatrix}$ if not. We make $R = 2000$ replications with $T = 2000$ observations. We generate 2000 more observations than required to eliminate initialization effects. We choose the true values of parameters to start the estimation procedure via ML and the unconditional variance for h_0 and ε_0^2. Both constrained and unconstrained optimization are applied. The code to generate the Monte-Carlo experiments is the following one:

1. `data = swgarch_sim(T,k,[0.1 0.1 0.92 ; 0.01 0.05 0.7],[0.9 0.1 ; 0.1 0.9],'NORMAL',ms_type,vald` with $T = 2000$, $k = 2$, `ms_type` is 1 or 2 depending on the experiment.

2. `[estimation, probabilities, diagnostics] = swgarch(data, k, 'NORMAL', ms_type, estim_cons, {'YES',[]}, [0.1 0.1 0.92 ; 0.01 0.05 0.7], [0.9 0.1 ; 0.1 0.9])` with `ms_type` is 'HAAS' or 'KLAASSEN', `ms_type` is 'CONS' or 'UNCONS' for constrained or unconstrained optimization, startvalDist is empty or 5.

3. store the results and repeat it R times.

Tables 3 and 4 show the results of the Monte-Carlo experiments. We report the mean of each parameter estimation, the asymptotic standard errors (A-StErr) and the mean squared errors (MSE). For both experiments, results show a good estimation of the true DGP. However, for the Klaassen's model, the γ_2 is underestimated. Augustyniak (2014) observes the same issue for Gray's model. This is due to the path dependence approximation proposes in their respective models.

File	Type	Usage	Description
Simulation			
`swgarch_sim.m`	Function	`swgarch_sim(args)`	Simulate different types of MS-GARCH processes with `k` states
`markovSim.m`	Function	`markovSim(args)`	Utility function used in `msGarchSim.m` to simulate a Markov chain
Estimation			
`swgarch.m`	Function	`swgarchest(args)`	Estimate different types of MS-GARCH models
`swgarch_parameters_check.m`	Function	`swgarch_parameters_check(args)`	Ensure that the input parameters are conformable (user inputs, stationarity conditions, sign of coefficients, ...).
`swgarch_starting_values.m`	Function	`swgarch_starting_values(args)`	If adequate starting values are user supplied, reformat as vectors. If not, returns the most decent one, randomly drawn, among a number of starting values filled by the user.
`swgarch_itransform.m`	Function	`swgarch_itransform(args)`	Inverse parameter transformation. Used to map parameters from the real line to a set of parameters appropriate for a MS-GARCH model. Utility function for an unconstrained estimation.
`swgarch_trans.m`	Function	`swgarch_trans(args)`	Used to map parameters from a MS-GARCH process to the positive unit simplex. Utility function for an unconstrained estimation
`swgarch_constr.m`	Function	`swgarch_constr(args)`	Compute constraints for constrained optimization

Table 1 List of the main files included in the MSG toolbox

File	Type	Usage	Description
`swgarch_likelihood.m`	Function	`swgarch_likelihood(args)`	Compute the likelihood of the specified MS-GARCH model
`swgarch_coreNK.m`	Function	`swgarch_coreNK(args)`	Core function for the estimation of a MSGARCH process of Klaassen (2002) with Gaussian errors
`swgarch_coreSTDK.m`	Function	`swgarch_coreSTDK(args)`	Core function for the estimation of a MSGARCH process of Klaassen (2002) with Student errors
`swgarch_coreNH.m`	Function	`swgarch_coreNH(args)`	Core function for the estimation of a MSGARCH process of Haas, Mittnik, et al. (2004) with Gaussian errors
`swgarch_coreSTDH.m`	Function	`swgarch_coreSTDH(args)`	Core function for the estimation of a MSGARCH process of Haas, Mittnik, et al. (2004) with Student errors
Utility functions			
`autocov.m`	Function	`autocov(args)`	Compute the empirical autocovariance function
`autocorr.m`	Function	`autocorr(args)`	Compute the empirical autocorrelation function
`robustvcv.m`	Function	`robustvcv(args)`	Compute a robust variance-covariance matrix
`pdf_kernel.m`	Function	`pdf_kernel(args)`	Compute the nonparametric density of a variable
`nwcov.m`	Function	`nwcov(args)`	Estimate the Newey-West covariance matrix
`hessien2sided.m`	Function	`hessien2sided(args)`	Estimate the hessian using two sided numerical derivatives
`matrixlag.m`	Function	`matrixlag(args)`	Compute the lag of a vector
`blokdiag.m`	Function	`blokdiag(args)`	Compute a block diagonal matrix

Table 2 List of the main files included in the MSG toolbox (continued)

	Value	Constrained			Unconstrained		
		Mean	A-StErr	MSE	Mean	A-StErr	MSE
α_{01}	0.01	0.013	0.004	0.000	0.013	0.004	0.000
α_{11}	0.05	0.074	0.062	0.005	0.075	0.062	0.005
γ_1	0.7	0.597	0.042	0.012	0.598	0.043	0.012
α_{02}	0.1	0.112	0.027	0.001	0.116	0.026	0.001
α_{12}	0.1	0.099	0.040	0.002	0.107	0.035	0.001
γ_2	0.92	0.931	0.065	0.004	0.913	0.045	0.002
p_{11}	0.9	0.898	0.014	0.001	0.899	0.020	0.001
p_{22}	0.9	0.898	0.017	0.001	0.897	0.023	0.001

Table 3 Experiment 1: simulation of a path-dependence MSGARCH and estimation of MSGARCH of Klaassen (2002).

	Value	Constrained			Unconstrained		
		Mean	A-StErr	MSE	Mean	A-StErr	MSE
α_{01}	0.01	0.010	0.002	0.000	0.010	0.002	0.000
α_{11}	0.05	0.053	0.015	0.000	0.052	0.014	0.000
γ_1	0.7	0.695	0.034	0.001	0.697	0.035	0.001
α_{02}	0.1	0.125	0.007	0.025	0.155	0.057	0.015
α_{12}	0.121	0.030	0.001	0.010	0.122	0.022	0.001
γ_2	0.92	0.886	0.013	0.003	0.881	0.024	0.001
p_{11}	0.9	0.898	0.130	0.000	0.898	0.120	0.000
p_{22}	0.9	0.898	0.160	0.000	0.898	0.160	0.000

Table 4 Experiment 2: simulation of a non path-dependence MSGARCH and estimation of MS-GARCH of Haas, Mittnik, et al. (2004).